APPARATUS FOR TESTING AN INTERCONNECTING LOGIC FABRIC

BACKGROUND

Technical Field

[0001] The present invention relates to systems and methods for Field Programmable Gate Arrays (FPGAs) and, more particularly, to systems and methods for testing FPGAs.

Related Art

[0002] Field Programmable Gate Arrays (FPGAs), Application Specific Integrated Circuits (ASICs) and microprocessor circuits are known, and have been used, according to performance requirements for a given circuit. For example, microprocessors are often preferred when flexibility and variable control are key design considerations. On the other hand, ASICs are often selected when performance or small circuit size is essential. FPGAs are often used when programmability and performance are important. Heretofore, however, FPGAs have typically been made to include only selectable logic blocks and have not included designs for robust processing of data. FPGAs have become very popular for telecommunication applications, Internet applications, switching applications, routing applications, etc.

[0003] Figure 1 illustrates a generic schematic block diagram of an FPGA 110. The FPGA 110 includes configurable logic fabric 112 (containing programmable logic gates and programmable interconnects) and configurable input/output blocks 114. The configurable input/output blocks 114 are fabricated on the perimeter of a substrate supporting the FPGA 110 and coupling to the pins of the integrated circuit to allow access to the configurable logic fabric 112.

[0004] The logic fabric 112 may be configured to perform a wide variety of functions corresponding to particular end user applications. For example, the configurable logic fabric 112 may be configured in a symmetric array

arrangement, a row-based arrangement, a column based arrangement, a hierarchical programmable logic device arrangement, or a sea-of-gates arrangement, each having different functional advantages.

[0005] Figure 2 illustrates the logic fabric 112 configured in a symmetrical array arrangement. Each logic block 216 of a plurality of logic blocks 216 is configured (usually by the end user) as an array of rows and columns to perform a specific logic function. More complex logic functions may be obtained by interconnecting individually configured logic blocks using a plurality of programmable interconnections 218. Accordingly, programmable interconnections 218 are formed between each of the logic blocks of each row and each column.

[0006] Programmable interconnections 218 also provide selective connectivity between each logic block of the array of logic blocks 216 and the configurable input/output blocks 114. Programmable interconnections 218 may be formed using static random access memory (RAM) cell technology, anti-fuse cell technology, EPROM transistor technology, and/or EEPROM transistor technology. If the FPGA utilizes static RAM programmable connections, the connections are made using pass transistors, transmission gates, and/or isolation circuits that are controlled by the static RAM cells.

[0007] If the FPGA utilizes anti-fuse interconnections, the interconnections typically reside in a high impedance state and can be configured into a low impedance state, or fused state, to provide the selective connectivity. If the FPGA utilizes EPROM or EEPROM based interconnections, the interconnection cells may be configured, thus allowing the FPGA to be reconfigured.

[0008] Figure 3 illustrates a schematic block diagram of the configurable logic fabric 112 being implemented in a row-based arrangement. In this configuration, the logic fabric 112 includes a plurality of logic blocks 216 arranged in rows. Between each row of the logic blocks are programmable

interconnections 218. Programmable interconnections 218 may be implemented utilizing static RAMs, dynamic RAMS and NVRAM, EPROM technology, and/or EEPROM technology.

[0009] Figure 4 illustrates a schematic block diagram of the logic fabric 112 being configured in a sea-of-gates configuration. The logic blocks and programmable interconnections are substantially similar to that described above.

[0010] Figure 5 illustrates the configurable logic fabric 112 being implemented as a hierarchical logic device. In this implementation, the configurable logic fabric 112 includes logic device blocks 522 and programmable interconnections 218. As shown, four logic device blocks 522 are in the corners with an interconnect 218 in the middle of the logic device blocks. In addition, the interconnects include lines coupling the configurable logic device blocks 522 to the interconnect 218. As such, the logic device blocks 522 may be configured to operate singularly or in combination with other logic blocks 522 according to the programming of the programmable interconnections 218.

[0011] As is known, field programmable gate arrays offer the end user the flexibility of implementing custom integrated circuits while avoiding the initial cost, time delay and inherent risk of application specific integrated circuits (ASIC). They also provide a degree of hardware-based customization that does not require custom application-specific designs, such as ASICs.

[0012] While FPGAs have these advantages, there are some disadvantages. For instance, an FPGA configured to perform a similar function as implemented in an ASIC, sometimes can require significantly more die area than the ASIC. The manufacturing expense of an FPGA, therefore, is greater than that of an ASIC. Additionally, FPGA performance is sometimes lower than that of an ASIC.

[0013] To mitigate some of the disadvantages of FPGAs with respect to ASICs, some FPGA manufacturers are including ASIC-

like functions on the same substrate as the configurable logic fabric. For example, FPGAs are now commercially available that include RAM blocks and/or multipliers in the configurable logic fabric 112. As such, the logic fabric 112 does not have to be configured to perform RAM functions and/or multiplier functions when such functions are needed. Thus, for these functions, significantly less die area is needed within the FPGA.

[0014] There are designs presently being developed to incorporate embedded microprocessors and other similar and known devices into an FPGA fabric by the present Assignee. As these designs mature, there will exist a need to provide for testing of the devices in a manner that enables one to determine whether the FPGA is formed and operating correctly, as well as, the processor or other device that is embedded there within.

[0015] Testers for testing integrated circuits are well known. Typically, a tester has local sequencers, each of which is programmable to establish operational logic states or a specified set of electrical conditions so that, with the input of data, an expected output may be compared to an actual output to determine proper operation of the device under test (DUT). In such systems, each local sequencer generates input data signals (events) for the DUT with reference to a global clock or other reference signals. Typically, sequencers are arranged and formed to present multiple test vectors to the (DUT). The sequencers further include memory and processing logic to provide the test vectors for testing the device.

[0016] Testers provide stimulus patterns for the DUT to prompt it to produce an expected output result with respect to the data transmitted to it for evaluation. Thereafter, the expected output is compared to an actual output to determine whether the DUT passed the test. In addition to testers, the use of scan latches to emulate pin connections is generally known. The scan latches are loaded with test

signals prior to a clock pulse being generated to prompt the device to process the information stored in the scan latches. A discussion of this technology in general terms may be found in the text Abramovici, Breuer and Friedman, *Digital System Testing and Testable Design*, (IEEE 1990).

[0017] The foregoing discussion of test vectors and test data patterns relates to devices for which pin access is not a problem. In an environment in which the device under test is embedded in a system, such as an FPGA fabric, providing stimulus patterns are not an achievable task without significant and, perhaps, unreliable, manipulation of the surrounding circuitry to produce the desired stimulus patterns and data inputs for testing the device. Accordingly, it is difficult to reliably and relatively easily provide the test vectors and test data to embedded devices within an FPGA fabric as part of running known test procedures.

[0018] What is needed, therefore, is a method and apparatus that enables one to test specific circuit components embedded within an FPGA by providing signals and measuring responses there from.

SUMMARY OF THE INVENTION

[0019] According to the present invention, test circuitry and operational methods that are formed within an FPGA device are to support the testing of embedded fixed logic core devices as well as fixed logic devices created within an interfacing logic portion that forms an interface between the embedded fixed logic core device and an FPGA fabric portion. The embedded fixed logic core device may be a processor or any other known fixed logic device. The test circuitry includes interconnect circuitry and test logic circuitry formed within the interfacing logic portion of the FPGA. Conceptually, the interfacing logic portion of the FPGA is often referred to as the "gasket" and is constructed in a manner similar to the construction (design and manufacture)

of ASICs. The inventive methods include configuring logic circuitry with the FPGA fabric portion to assist in testing the fixed logic core device as well as the fixed logic devices formed within the gasket. During testing operations, the FPGA fabric is configured to form scan latches to become a scan chain that will receive test vectors. The test vectors are applied via the test circuitry within the gasket to the input/output pins or ports of the fixed logic core device or to the fixed logic device formed within the gasket that is being tested.

[0020] The test circuitry within the gasket further includes isolation circuit elements that allow the FPGA fabric (as configured for testing) to have direct access to the input/output of the device under test whether it is the fixed logic device formed within the gasket or an embedded core processor. An FPGA fabric that is configured for testing can perform structural tests for a specific device because the necessary access to the device inputs and outputs are provided, at least in part, by test circuitry formed within the gasket. Because test vectors or signals may be provided to the embedded device(s) as well as the fixed logic circuitry, structural testing of the entire FPGA is supported thereby enabling a tester to determine with a large degree of certainty whether the FPGA is operational.

[0021] In the described embodiment of the invention, the logic within the FPGA fabric that may be configured for testing may also be configured for normal operation while the testing is not taking place. In an alternate embodiment, however, logic circuitry is formed within the FPGA fabric exclusively for test processes and purposes. The manner in which the logic of the FPGA may be configured for test or normal operation is known by one of the ordinary skill in the art.

[0022] In the described embodiments of the invention, interfacing logic (gasket logic) interfaces the embedded fixed logic core device to the FPGA fabric as is described

herein. While the testing of the FPGA fabric is a generally known task, testing of the gasket logic and embedded core devices is not known. Thus, according to a second aspect of the present invention, the gasket logic includes specialized testing configurations that facilitate testing of the gasket logic as well as the embedded core. There potentially are thousands of different configurations that correspond to the specific device under test.

[0023] As was the case with the testing of the fixed logic core device, the FPGA fabric is also configurable to test the gasket logic. According to one embodiment of a testing operation, the FPGA fabric is configured to generate scan chains that may be used to generate test vectors for testing the gasket logic. The FPGA fabric is also configured to receive output vectors produced by the gasket logic and to evaluate the test results by comparing these output vectors to expected results.

[0024] The output test vectors, in the described embodiments of the invention, are evaluated differently according to whether they originated from the embedded fixed logic device or the gasket logic (interfacing logic). Test vectors output from the embedded fixed logic device are produced to an external system for evaluation (e.g., a tester) while the output test vectors from the gasket logic are evaluated on the chip itself by logic within the FPGA fabric. In an alternate embodiment, however, the test vectors from the FPGA fabric also are evaluated externally from the chip. In yet another embodiment, all output test vectors are evaluated on chip.

[0025] The present invention is advantageous in that it solves the problem of how to test an embedded device as well as the interfacing logic between the embedded device and the fabric portion of an FPGA. Forming scan chains with test vectors, for example, to test an ASIC (e.g., the gasket logic and its embedded components), as well as to test an embedded device that is embedded within an ASIC, not only solves

connection issues, but also makes it possible to develop an embedded system, as in the described embodiments of the invention. Other aspects of the present invention will become apparent with further reference to the drawings and specification, which follow.

BRIEF DESCRIPTION OF THE DRAWINGS

[0026] A better understanding of the present invention can be obtained when the following detailed description of the preferred embodiment is considered with the following drawings, in which:

[0027] Figure 1 illustrates a schematic block diagram of a prior art field programmable gate array;

[0028] Figure 2 illustrates a schematic block diagram of the configurable logic fabric of the programmable gate array of Figure 1 configured in a symmetrical array;

[0029] Figure 3 illustrates a schematic block diagram of the logic fabric of the programmable gate array of Figure 1 in a row based configuration;

[0030] Figure 4 illustrates a schematic block diagram of the logic fabric of the programmable gate array of Figure 1 in a sea-of-gates configuration;

[0031] Figure 5 illustrates a schematic block diagram of the logic fabric of the programmable gate array of Figure 1 in a hierarchical programmable logic device configuration;

[0032] Figure 6 illustrates a block diagram of a programmable gate array in accordance with the present invention;

[0033] Figure 7 illustrates a graphical diagram of an alternate programmable gate array in accordance with the present invention;

[0034] Figure 8 illustrates a graphical diagram of another programmable gate array in accordance with the present invention;

[0035] Figure 9 illustrates a more detailed graphical diagram of a portion of the programmable gate array of an

embedded device of Figure 6;

[0036] Figure 10 illustrates a schematic block diagram of a microprocessor embedded in an FPGA in accordance with the present invention;

[0037] Figure 11 is a flow chart illustrating a method for testing fixed logic circuitry within the interfacing logic according to one embodiment of the present invention;

[0038] Figure 12 illustrates a schematic block diagram of a few of the interconnecting tiles operably coupled to the surrounding programmable logic fabric;

[0039] Figure 13 is a flow chart illustrating a method for testing fixed logic circuitry within the interfacing logic according to one embodiment of the present invention;

[0040] Figure 14 is a flow chart illustrating a method for assigning a permanent ID to a device according to one aspect of the present invention;

[0041] Figure 15 illustrates a functional diagram of yet another programmable gate array in accordance with the present invention;

[0042] Figure 16 illustrates a functional diagram of a variation of the programmable gate array of Figure 15;

[0043] Figure 17 illustrates a functional diagram of a further variation of the programmable gate array of Figure 15.

[0044] Figure 18 is a functional block diagram illustrating an FPGA formed according to one embodiment of the present invention;

[0045] Figure 19 is a block diagram generally illustrating the components of a fixed logic core processor block that is embedded within an FPGA fabric according to the present invention;

[0046] Figure 20 is a functional block diagram illustrating the connectivity between various FPGA and fixed logic core processor blocks constructed according to the present invention;

[0047] Figures 21 and 22 are functional block diagrams

illustrating functional blocks of an FPGA having an embedded fixed logic core, gasket logic, and test circuitry, selectively configurable according to different test modes of operation;

[0048] Figure 23 is a functional block diagram illustrating a circuit configuration that supports selective ID Assignment according to one embodiment of the present invention;

[0049] Figure 24 is a functional block diagram illustrating a circuit in which scan data is scanned into and out of a scan chain to test a logic block;

[0050] Figure 25 is a functional block diagram illustrating a variety of test modules configured within an FPGA fabric and how they are used to test gasket logic according to the present invention;

[0051] Figure 26 illustrates a portion of CRC circuitry used to receive the outputs from the gasket logic and to provide the output for comparison purposes;

[0052] Figure 27 is a functional block diagram illustrating the manner in which clocks, input/output vectors, and scan test vectors are applied to test a fixed logic core that is embedded within an FPGA fabric according to the present invention;

[0053] Figure 28A illustrates a double latch shift register that is used in one embodiment of the logic blocks of the input configuration 2702 of Figure 27;

[0054] Figure 28B is a timing diagram illustrating the timing of the application of various clock signals according to the method for testing a fixed logic core according to one embodiment of the present invention;

[0055] Figure 29 is a functional block diagram illustrating the use of a scan chain in an FPGA fabric to test an embedded device according to on aspect of the present invention;

[0056] Figure 30 is a functional schematic diagram of an FPGA constructed according to one embodiment of the invention

that includes a plurality of embedded devices;

[0057] Figure 31 is a flow chart illustrating a method for testing a device that is embedded within fixed logic circuitry that is itself at least partially embedded within an FPGA according to one embodiment of the invention;

[0058] Figure 32 is a flow chart illustrating a method for testing an embedded device that is embedded within an ASIC that is within an FPGA according to one embodiment of the invention; and

[0059] Figure 33 is a flow chart illustrating a method for testing a module embedded within an ASIC that, in turn, is embedded in an FPGA according to one embodiment of the present invention.

DETAILED DESCRIPTION OF THE DRAWINGS

[0060] Generally, the present invention provides interconnecting logic that interfaces an embedded fixed logic circuit, or circuits, with configurable logic fabric of a field programmable gate array. The interconnecting logic enables any fixed logic circuit (e.g., a digital signal processor, microprocessor, physical layer interface, link layer interface, network layer interface, audio processor, video graphics processor, and/or applications-specific integrated circuit) to be embedded within the configurable logic fabric of a field programmable gate array. In addition, the interconnecting logic provides connectivity between the fixed logic circuit and the configurable logic fabric such that the fixed logic circuit functions are an extension of the configurable logic fabric.

[0061] The interconnecting logic includes interconnecting tiles and may further include interfacing logic. The interconnecting tiles provide selective connectivity between inputs and/or outputs of the fixed logic circuit and the interconnects or programmable interconnections of the configurable logic fabric. The interfacing logic, when activated to be electrically present, provides logic

circuitry that conditions data transfers between the fixed logic circuit and the configurable logic fabric. The conditioning of the data may be format changes, parallel-to-serial conversion, serial-to-parallel conversion, multiplexing, de-multiplexing, performing Boolean logic functions, etc. With such interconnecting logic, any fixed logic circuit may be readily embedded within a programmable gate array to provide additional functionality to the end users of FPGAs.

[0062] The present invention can be more fully described with reference to Figures 6 through 33. Figure 6 illustrates a block diagram of a field programmable gate array 300. The field programmable gate array 300 includes the configurable logic fabric 112, the configurable input/output blocks 114, a fixed logic circuit 632 and interconnecting logic 634. The fixed logic circuit 632 may be a digital signal processor, microprocessor, physical layer interface, link layer interface, network layer interface, audio processor, video graphics processor, logic circuitry, and/or applications-specific integrated circuits.

[0063] Typically, the fixed logic circuit 632 includes a plurality of inputs and a plurality of outputs, which are represented by input/output ports 636, 638, 640 and 642. The input/output ports 636-642 are operably coupled to the interconnecting logic 634, which provides connectivity between the input/output ports of the fixed logic circuit 632 with the configurable logic fabric 112 of the FPGA 300. It should be noted that more than one fixed logic circuit can be included in the programmable gate array.

[0064] The configurable logic fabric 112 includes a plurality of configurable logic blocks (CLBs) and programmable interconnects. The architecture of the configurable logic fabric may be row or column based, hierarchical-PLD, symmetrical array, and/or a sea-of-gates. The configurable logic blocks, interconnects, and I/O blocks may be, for example, of the type manufactured and distributed

by Xilinx, Inc. The interconnects may include a plurality of switch matrix devices that utilize static RAM cell technology, anti-fuse cell technology, EPROM transistor technology, and/or EEPROM transistor technology.

[0065] The field programmable gate array 300 may be implemented as an integrated circuit wherein the configurable I/O blocks 114, configurable logic fabric 112, the interconnecting logic 634 and the fixed logic circuit 632 are fabricated on a substrate. In one embodiment, the circuitry of each of these elements 112, 114, 632 and 634, are implemented using CMOS technology on a silicon substrate. However, as one of average skill in the art will appreciate, other integrated circuit technologies and substrate compositions may be used.

[0066] In operation, the interconnecting logic 634 provides coupling between the configurable logic fabric 112 and the fixed logic circuit 632. As such, end users of the field programmable gate array 300 may program it by treating it as a component of the configurable logic fabric 112. For example, if the fixed logic circuit 632 includes a microprocessor, the interconnecting logic 634 may include memory for storing programming instructions and data in addition to connectivity to memory of the FPGA 300.

[0067] Accordingly, the configurable logic fabric 112 is configured to perform desired functions in combination with the fixed logic functions of the microprocessor. Thus, with an embedded microprocessor, the field programmable gate array 300 offers the flexibility of an FPGA, with the processing efficiency of an application-specific integrated circuit microprocessor. In addition, by embedding a microprocessor within the configurable logic fabric 112, as opposed to having two separate integrated circuits (one for the microprocessor and another for the FPGA), power consumption is reduced due to the elimination of interconnecting pins and traces between the two separate integrated circuits. Further, the field programmable gate array 300 requires less

printed circuit board real estate than separate integrated circuits for an FPGA and a microprocessor.

[0068] Figure 7 illustrates a graphical diagram of an alternate field programmable gate array 500. The field programmable gate array 500 includes the configurable logic fabric 112, the configurable input/output blocks 114, a 1st fixed logic circuit 632, 1st interconnecting logic 634, a 2nd fixed logic circuit 752 and 2nd interconnecting logic 754. In this illustration, the 1st interconnecting logic 634 and 1st fixed logic circuit 632 are as generally described with reference to Figure 6.

[0069] The 2nd fixed logic circuit 752 may include any logic functions, including those of a digital signal processor, microprocessor, physical layer interface, link layer interface, network layer interface, audio processor, video graphics processor, logic circuitry, and/or an application-specific integrated circuit. The 2nd fixed logic circuit 752 includes a plurality of input/output ports 756, 758, 760 and 762 that allow it to interface with the 2nd interconnecting logic 754. The 2nd interconnecting logic 754 provides the connectivity between the 2nd fixed logic circuit 752 and the configurable logic fabric 112.

[0070] Figure 8 illustrates a graphical diagram of another field programmable gate array 700. The field programmable gate array 700 includes the configurable logic fabric 112, the configurable input/output blocks 114, and four fixed logic circuits 632, 752, 872 and 876. The structure of each fixed logic circuit is similar to the fixed logic circuit shown in Figure 7 (note that the I/Os in each fixed logic circuit are not shown because of the limited size of the drawings). Each fixed logic circuit 632, 752, 872 and 876 has its own corresponding interconnecting logic 634, 754, 874 and 878, respectively. The interconnecting logic 634, 754, 874 and 878 provide its respective fixed logic circuit connectivity to the configurable logic fabric 112.

[0071] The construct of interconnecting logic 634, 754,

874 and 878 will be dependent upon the type of fixed logic circuit it is supporting. For instance, if the logic circuit is a fixed logic function, the interconnecting logic 634, 754, 874 and/or 878 would include interconnecting tiles in order to perform a particular Boolean function. The interconnecting tiles will be described in greater detail with reference to Figures 9, 10 and 12. If, however, the fixed logic circuit is more complex, such as a digital signal processor, microprocessor, physical layer interface, link layer interface, network layer interface, audio processor, video graphics processor, and/or applications-specific integrated circuit, the interconnecting logic 634, 754, 874 and/or 878 will include a plurality of interconnecting tiles and interfacing logic. The interfacing logic will be described in greater detail with reference to Figures 9 and 10.

[0072] Figure 9 illustrates a more detailed graphical diagram of a portion of the field programmable gate array 300 with an embedded device of Figure 6. While Figure 9 is illustrated with reference to the FPGA 300 of Figure 6, the concepts regarding the interconnecting logic 634 is equally applicable to the interconnecting logic 754 of Figure 7, and the interconnecting logic 754, 874, and 878 of Figure 8. As one of average skill in the art will appreciate, any number of fixed logic circuits may be embedded within the configurable logic fabric using interconnecting logic.

[0073] As shown in Figure 9, the configurable logic fabric 112 includes a plurality of configurable logic blocks (CLBs) 980, a plurality of block random access memory (RAM) 990, and a plurality of multipliers 992. The configurable I/O block section 114 shown on Figure 1 includes a plurality of individual I/O blocks (IOB) 986 and a plurality of digital clock managers (DCM) 984. The operations of the configurable logic blocks 980, the digital clock managers 984, the input/output blocks 986, the block RAM 990, and the multipliers 992 function in a similar manner as corresponding

components found in the family of field programmable gate arrays designed and manufactured by Xilinx, Inc.

[0074] As shown, the configurable logic blocks 980, the block RAM 990 and the multipliers 992 are arranged in a series of rows and columns. The fixed logic circuit 632 displaces some of the components in programmable logic fabric 112. As such, the fixed logic circuit 632 and the interconnecting logic 634 replace a set of configurable logic blocks 980, a set of memory blocks 990, and/or a set of multipliers 992.

[0075] Figure 10 illustrates a schematic block diagram of a microprocessor 1000 being embedded in the FPGA 300 as an example of a fixed logic circuit. It should be noted that the present invention is applicable to processors of any design, and is not limited to a particular type of processor. As one of average skill in the art will appreciate, the physical design of the microprocessor 1000 can have a variety of geometric configurations. The microprocessor 1000 is surrounded by the interconnecting logic 634 (shown in Fig. 9) that includes the interfacing logic 994 and a plurality of interconnecting tiles 996. The microprocessor 1000 may be connected to block RAMs 990 through memory controllers (not shown). The microprocessor 1000 may be directly connected to the block RAMs 990. By providing coupling between the microprocessor 1000 and the block RAMs 990, the block RAMs 990 may be shared by the microprocessor 1000 and the programmable logic fabric 112 of Figure 1. Such direct sharing eliminates the need for programming the programmable logic fabric to provide the microprocessor with access to the block RAMs 990.

[0076] The interfacing logic 994 may contain one or more blocks of logic gates 1014. These blocks may be designed to perform any logic function, and may communicate in any manner with the microprocessor 1000, the block RAMs 990, and the interconnecting tiles 996. In Figure 10, only one such block of logic functions is shown. The interfacing logic 994 may

also contain one or more blocks of configurable logic gates 1016. These blocks may be configured to perform any logic function, and may communicate in any manner with the microprocessor 1000, the block RAMs 990, and the interconnecting tiles 996. In Figure 10, only one such block of configurable logic functions is shown. The interfacing logic 994 may further contain a test module 1003 that controls the manufacturing testing of the microprocessor 1000, interconnecting tiles 996, and/or various parts of the interfacing logic 994. In Figure 10, even though the test module 1003 is shown as an isolated block to simplify the diagram, in reality it would be connected to some or all of the above-mentioned components. A control module 1005 can be used to control the operations of the microprocessor 1000 and various components in the interfacing logic 994. The interfacing logic 994 may also contain a timing module 1007 that generates various timing signals for the microprocessor 1000 and other components in the interfacing logic 994. The timing module 1007 may contain clock generation circuits (such as oscillators), or may use some of the clock signals of the programmable logic fabric. In Figure 10, even though the control module 1005 and timing module 1007 are shown as isolated blocks, they are in reality connected to some or all of the above-mentioned components. In addition, modules performing other functions may also be included.

[0077] The microprocessor 1000 may communicate directly with the interconnecting tiles 996 (which are programmably connected to the CLBs 980 shown in Figure 9). The microprocessor 1000 may also communicate with the interconnecting tiles 996 through the blocks of logic gates 1014 and blocks of configurable logic gates 1016. The connections shown in Figure 10 could be unidirectional and/or bidirectional.

[0078] The block RAMs 990 may store at least a portion of the executable instruction code for the microprocessor 1000. In addition, such memory may store the data to be processed

by the microprocessor 1000 and the data already processed by the microprocessor 1000. Because the memory is shared between the microprocessor 1000 and the programmable logic fabric 112, configured portions of the programmable logic fabric 112 may retrieve the data to be processed and/or the data already processed to perform a certain function upon the data. It should be noted that the block RAMs 990 may be at any position relative to the microprocessor 1000 (top, down, left or right).

[0079] Figure 11 is a flow chart illustrating a method for testing fixed logic circuitry within the interfacing logic according to one embodiment of the present invention. Generally, the methods relating to performing test includes utilizing the inventive structure described herein. More specifically, the method includes configuring the FPGA into a test mode (step 1104). Thereafter, test signals are transmitted to the fixed logic circuitry within the interfacing logic from the FPGA fabric (whether generated there or externally (step 1108). Thereafter, test signal outputs from the fixed logic circuitry are transmitted to a test multiplexer (step 1112). The test signal outputs are then transmitted from the multiplexer through the fixed interfacing logic to the FPGA fabric (step 1116). In one embodiment of the invention, the FPGA fabric is configured to analyze the test signal outputs received from the multiplexer to determine whether the fixed logic circuitry passed or failed the test. In another embodiment of the invention, the test signal outputs are merely conducted through the FPGA fabric to an external device, or system, such as a tester, to determine pass or fail.

[0080] Figure 12 illustrates a schematic block diagram of a few of the interconnecting tiles 996-1 through 996-6 operably coupling to the surrounding programmable logic fabric. The surrounding programmable logic fabric includes a plurality of configurable logic elements (CLE) 1280-1 through 1280-13 and corresponding programmable switch matrices 1252

through 1278. Solid lines between the programmable switch matrices represent various interconnect lines that provide connectivity in the programmable logic fabric. An example of an FPGA architecture that can be used in the present invention can be found in U.S. Patent No. 5,914,616 entitled, "FPGA Repeatable Interconnect Structure with Hierarchical Interconnect Lines."

[0081] Each interconnecting tile 996 contains a programmable switch matrix that is programmably connected to (a) a programmable switch matrix in the programmable logic fabric, (b) a termination tile (called herein "term tile"), and (c) adjacent interconnecting tiles. Figure 12 shows six switch matrices labeled 996-1-s to 996-6-s in the interconnecting tiles 996-1 to 996-6, respectively. As an example, the switch matrix 996-2-s is connected to the switch matrix 1254 in the programmable logic fabric, a term tile T2, and adjacent switch matrices 996-1-s and 996-3-s. Similarly, the switch matrix 996-5-s is connected to the switch matrix 1264 in the programmable logic fabric, a term tile T4, and adjacent switch matrices 996-4-s and 996-6-s. The six programmable switch matrices 996-1-s to 996-6-s each contains a plurality of connections (shown as lines 1251-1 to 1251-6, respectively) that are connected to the microprocessor 1000 and/or components in the interfacing logic 994.

[0082] The structure of switch matrices 996-1-s to 996-6-s is substantially the same as that of the switch matrices in the programmable logic fabric.

[0083] The function of the term tiles is to terminate the interconnect lines and/or provide connectivity to the lines that are interrupted by the microprocessor 1000 and/or components of the interfacing logic 994. In one embodiment (e.g., the FPGA described in the above mentioned U.S. Pat No. 5,914,616), the programmable logic fabric contains single, hex and long lines. In the term tiles, the single lines are U-turned to other single lines, the hex lines are rebuffered and span to the far side of the microprocessor 1000, and the

long lines span the microprocessor 1000.

[0084] Figure 13 is a flow chart illustrating a method for testing fixed logic circuitry within the interfacing logic according to one embodiment of the present invention. Generally the method includes configuring the FPGA into a test mode (step 1304). Thereafter, test signals are transmitted to a test multiplexer within the interfacing logic (step 1308). The test signals are then transmitted from the test multiplexer to fixed logic circuitry (either an embedded device or fixed logic formed within the gasket or interfacing circuitry (step 1312). The test signal outputs from the fixed logic circuitry are then transmitted through a communication path formed within the gasket (interfacing logic) to the FPGA fabric (step 1316). In one embodiment of the invention, the FPGA fabric is configured to analyze the test signal outputs to determine whether the embedded core device passed or failed the test. In another embodiment of the invention, the test signal outputs are merely conducted through the FPGA fabric to an external device, or system, such as a tester, to determine pass or fail.

[0085] Figure 14 is a flow chart illustrating a method for assigning a permanent ID to a device according to one aspect of the present invention. As discussed previously, it is advantageous to have a system in which inputs for receiving a device ID may be stimulated during test procedures. Typically, however, approaches that facilitate this capability store the device ID in memory. Through any one of a plurality of situations, however, an ID may be overwritten with an older and presently invalid ID. Accordingly, the safest solution heretofore has been to not provide the desirable capability of stimulating the inputs for receiving the permanent ID. The invention illustrated in Figure 14 provides a way of stimulating ID inputs during testing without using a scheme in which the device ID is stored in memory and can be overwritten.

[0086] More specifically, the invention includes a

plurality of steps, the first of which is to generate a permanent ID for a device (step 1404). Typically, such an ID is defined by logic or hardware of the FPGA. The ID is then transmitted to a multiplexer (step 1408). Then, under non-test conditions, the ID is transmitted to a device to enable it to know its own ID while it is operating (step 1412).

[0087] In test mode, however, the multiplexer receives an indication that it is in test mode (step 1416). With this indication, the multiplexer changes the coupling to decouple the device from the hardware that defines the device ID (the FPGA in the described embodiment of the invention). The device is then coupled to a test signal generation source. The test signal generation source may be one of many different devices, including, for example, logic within the FPGA, fixed logic circuitry formed within the FPGA or an external tester.

[0088] Thus, once the multiplexer has performed it's switching, it then receives test data for delivery to the inputs of the device that normally receive the permanent device ID (step 1420). The test data is transmitted from the multiplexer to the device (step 1424). Once testing is complete, the multiplexer receives an indication of Normal operation (step 1428) and switches the connections back to the original configuration and resumes transmitting the permanent device ID to the device (step 1432).

[0089] Figures 15, 16 and 17 are functional schematic diagrams illustrating a plurality of arrangements of a fixed logic device formed or "cut into" a configurable logic fabric in which the embedded device includes interconnecting logic between it and the configurable logic fabric according to various embodiments of the present invention. As may be seen in Figure 15, a high speed data interface 1532 is surrounded by a 1st interconnecting logic 1536 while a fixed logic processing module 1534 is surrounded by a 2nd interconnecting logic 1538. Figure 16 is similar to Figure 15 with the exception that the high-speed data interface 1632 is placed

at the edge of the configurable logic fabric. Accordingly, the 1st interconnecting logic 1636 is only formed to surround the high-speed data interface 1632 on those sides that are bound by the configurable logic fabric. Figure 17 is similar to Figure 16 with the exception that the high-speed data interface is placed within a corner of the configurable logic fabric 112. Accordingly, the 1st interconnecting logic 1736 is formed on only two sides of the high-speed data interface 1732.

[0090] Figure 18 is a functional block diagram illustrating an FPGA formed according to one embodiment of the present invention. The FPGA 1800 includes a fixed logic core processor block 1804 that is embedded within the FPGA fabric. In the illustrated embodiment, the fixed logic core processor block 1804 includes a microprocessor, e.g., Power PC 405, or another type of fixed logic device as was previously described herein. The fixed logic core processor block 1804 is fabricated so that a programmable FPGA fabric 1800 surrounds it. The FPGA fabric includes CLBs, block RAM, interconnections, etc. With this construction of the fixed logic core processor block 1804, no direct coupling exists between the input/output of the FPGA 1800 and the fixed logic core processor block in the embodiment shown. All external coupling to the fixed logic core 1804 must traverse the FPGA fabric.

[0091] Figure 19 is a block diagram generally illustrating the components of the fixed logic core processor block 1804 of Figure 18. The core processor block 1804 includes gasket logic 1904 and fixed logic core processor 1908. Gasket logic 1904 interfaces the fixed logic core processor 1908 with the FPGA fabric within which it is embedded. The gasket logic 1904 includes a pair of On Chip Memory controller modules (OCMs) 1912, a controller 1916, and additional logic.

[0092] The OCMs 1912 interface the fixed logic core processor 1908 with block RAM of the FPGA fabric. The controller 1916 is accessible by the FPGA fabric to allow the

FPGA fabric to control the gasket logic 1904 and the fixed logic core processor 1908. The gasket logic 1904 is fixed logic circuitry wherein its primary function is to interface the fixed logic core processor 1908 with the FPGA fabric. Many aspects of the FPGA fabric have already been described herein with reference to Figures 6-17.

[0093] Continuing to refer to Figure 19, the gasket logic also includes a plurality of multiplexer arrays 1920 that multiplex data, address, and control lines between fixed logic core processor 1908, OCMs 1912, and controller 1916. These multiplexer arrays 1920 also serve to: (1) isolate the fixed logic core processor 1908 from the gasket logic 1904 during testing of the fixed logic core processor 1908 and (2) isolate the gasket logic from the fixed logic core processor 1908 from the gasket logic 1904 during testing of the gasket logic 1904. As will be further described herein, during testing of the fixed logic core processor 1908 and gasket logic 1904, the FPGA fabric and the multiplexer arrays 1920 will be operated to apply scan vectors and other test inputs to the fixed logic core processor 1908 and to the gasket logic 1904.

[0094] Figure 20 is a functional block diagram illustrating the functional connectivity between various FPGA and fixed logic core processor blocks according to one described embodiment of the present invention. The components are laid out in a functional manner to illustrate how they may be coupled for testing purposes. An FPGA gasket 2004 is formed within the FPGA 2000 and includes interfacing circuitry 2012 and Block RAM 2008.

[0095] During testing of the fixed logic core processor 2016 (an embedded device; for example, a Power PC), the FPGA fabric 2000 is configured to access, stimulate, and receive outputs from the fixed logic core processor 2016. Likewise, during testing of the gasket logic 2020, the FPGA fabric 2000 is configured to access, stimulate, and receive output from the gasket logic 2020.

[0096] More particularly, during testing of the fixed logic core processor 2016, the FPGA fabric 2000 is configured to provide access to scan chains of the fixed logic core processor 2016, scan in test vectors to the fixed logic core processor 2016, and scan out results from the fixed logic core processor 2016. Further, during the testing of the fixed logic core processor 2016, the FPGA fabric 2000 is configured to stimulate the input of the fixed logic core processor 2016 and to receive outputs produced by the fixed logic core processor 2016. The manner in which fixed logic core processors 2016 are tested using a test socket and a coupled tester is generally known. However, use of the FPGA fabric 2000 and multiplexers 1920 (of Figure 19) to test the fixed logic core processor 2016 is unique to the present invention.

[0097] Figures 21 and 22 are functional block diagrams illustrating functional blocks of an FPGA having an embedded fixed logic core processor, gasket logic, and test circuitry, selectively configurable according to different test modes of operation. During testing of the fixed logic core processor 2116, FPGA fabric 2104 is configured according to the test mode of operation (gasket test or embedded core device test). Test circuitry formed within FPGA 2104 comprises logic circuitry that is configured for test whenever the FPGA is in a test mode of operation and is constructed using known FPGA design and fabrication techniques. In one embodiment of the invention, the logic circuitry of FPGA 2104 comprises a configuration for operation in a first test mode in which portions of the gasket logic 2108 and 2128 are tested and a second configuration for operation in a second test mode in which the embedded fixed logic core device is tested.

[0098] The logic portions are shown in Figures 21 and 22 as a plurality of modules shown generally at 2124. These modules 2124 provide inputs to the fixed logic core processor 2116, receive outputs from the fixed logic core processor 2116, and support the interface of the FPGA (fabric, fixed

processing core, gasket logic, etc.) with a tester via FPGA input/output pins. Thus, these modules 2124 emulate a portion of the testing functions that would otherwise be implemented by the tester. These testing functions allow the fixed processing core 2116 to be tested as if all of its input/output and control pins were externally accessible.

[0099] Figure 21 also illustrates the configuration of the test circuitry while testing an embedded fixed logic core processor. In reference to Figures 21 and 22, steady state signals are indicated with the symbol "+-". Additionally, dashed lines represent communication lines that are not carrying test data.

[0100] During testing of the embedded fixed logic core processor 2116, the FPGA fabric 2104, while configured in a test mode of operation, uses data path 2172 to access the inputs of fixed logic core processor 2116 and uses path 2168 to observe the outputs of the fixed logic core processor 2116. The data paths 2172 and 2168 are a plurality of bits wide, the width of these data paths depending upon the number of inputs and outputs of the fixed logic core processor 2116.

[0101] Multiplexer 2120 allows the FPGA fabric 2104 to directly access the fixed logic core processor inputs 2140. FPGA 2104, when configured in a test mode for testing an embedded core device, activates the multiplexer 2120 to couple data path 2172 to inputs of core 2116. Further, the FPGA fabric 2104 observes the outputs 2152 of the fixed logic core processor 2116 via data path 2168. During testing of the embedded fixed logic core processor 2116, the FPGA fabric 2104 also may provide additional inputs via data paths 2144 and observe additional outputs via data path 2148.

[0102] While the test circuitry is configured to test the fixed logic core processor 2116, multiplexer 2132 is controlled to prevent the gasket logic 2128 from receiving the outputs on communication path 2152 from the fixed processor core 2116. In such case, the application of steady state input signals by the FPGA fabric 2104 via data path

2176 hold the gasket logic 2128 in a known state. Similarly, the FPGA fabric 2104 may hold the gasket logic 2108 in a known state via application of steady state input data signals via data path 2112.

[0103] Figure 22 illustrates the configuration of the test circuitry while testing the gasket logic 2108 and 2128. During testing of the gasket logic 2108, the FPGA fabric 2104 uses data path 2112 to apply test signals to the gasket logic 2108 and uses data path 2164 to observe the output of gasket logic 2108. These data paths 2112 and 2164 are a plurality of bits wide, the width of these data paths depending upon the number of inputs and outputs of the gasket logic 2108. While the test circuitry is configured to test the gasket logic 2108, multiplexer 2120 may be controlled to apply a known state to the fixed processor core 2116. In such case, the application of steady state input data by the FPGA fabric 2104 via data path 2172 and 2140 would hold the fixed processor core 2116 in a fixed state.

[0104] Further, while testing the gasket logic 2128, multiplexer 2132 allows the FPGA fabric 2104 to directly access the inputs 2156 of the gasket logic 2128 via data path 2176. The FPGA fabric 2104 observes, therefore, the outputs 2160 of the gasket logic 2128 directly. Gasket logic 2128 receives test input signals on path 2156 that are generated by FPGA 2104 onto path 2176 through multiplexer 2132. As is understood, multiplexer 2132 couples path 2176 to path 2156 responsive to a control signal generated by FPGA fabric portion 2104. For simplicity, the control signals for the multiplexers are not shown herein.

[0105] Figure 23 is a functional block diagram illustrating a circuit configuration that supports selective ID Assignment according to one embodiment of the present invention. An FPGA or other logic circuit is formed to generate an assigned ID that is used for transmission to a core device. Herein, an ID Assignment module 2304 generates a permanently stored ID to core device 2308. Often, for test

purposes, it is desirable to transmit test signals in place of the core ID to the core during testing. One approach is to store an ID value in a non-permanent manner in volatile memory within an ID Assignment module. The transmission of the ID, therefore, may be driven by software and may be replaced by test signals during the test modes of operation.

[0106] One problem with this approach, however, is that voltage surges and other similar events can corrupt the ID values stored in the volatile memory. Accordingly, if an ID is impressed upon any type of ordinarily non-volatile memory to avoid undesired loss of the most current device ID, then test signals cannot be applied to the core device at the inputs that usually received the device ID.

[0107] Figure 23 illustrates a solution to the aforementioned problem. A test data generator 2312 is coupled to the input side of a plurality of multiplexer units 2316-2328. More specifically, the test data generator typically includes at least one line that is coupled to each multiplexer 2316-2328 for producing test data thereto. Another input of each multiplexer 2316-2328 is coupled to receive a data line that is used for carrying an ID portion. The ID portion is stored within non-volatile memory of ID Assignment module 2304. Thus, there is one multiplexer unit for each data line that is used for carrying an ID portion. Accordingly, the default setting of the multiplexers 2316-2328 couples the ID Assignment module 2304 to the Core 2308 to facilitate delivery of its device ID.

[0108] During test, however, test data generator 2312 also is coupled to produce control signals to multiplexers 2316-2328 to prompt them to decouple the ID Assignment Module 2304 and to couple each of a plurality of test data lines that are coupled to the input side of the multiplexers 2316-2328 to core 2308. The described invention of Figure 23 is advantageous in that an ID may be coded or embedded into hardware logic while maintaining the ability of a test data generator to produce test data to the input pins of a core

device under tests that normally are for receiving ID values.

[0109] Figure 24 is a functional block diagram illustrating a circuit in which scan data is scanned into and out of a scan chain to test a logic block. As was previously described with reference to Figures 21 and 22, the gasket logic 2108 and 2128 and the fixed logic core processor 2116 must be tested after fabrication. Further, as was described, testing of the gasket logic 2108 and 2128 is performed by applying inputs to the gasket logic 2108 and 2128, receiving the output produced by the gasket logic 2108 and 2128 in response to the applied input, and comparing the output produced to expected output. As was further described, some of the functionality required for this testing is performed by configuring the FPGA fabric 2104 and operating the configured FPGA fabric 2104 to perform this testing. Additionally, as has been described, scan chains may be used to stimulate the logic being tested with test vectors.

[0110] Generally speaking, scan chain testing includes placing the device 2404 or circuit under test into a known state using configurations in the FPGA fabric. Test vectors are loaded into the scan chains 2412 and then are produced to the fixed logic device 2404 as inputs (applying particular inputs as stimulus), clocking the device 2404 for one or more clock cycles and then receiving and analyzing outputs states of the fixed logic processor. The test results may also be stored into the scan chain and be produced externally for evaluation.

[0111] More specifically, the manner in which the FPGA fabric may be configured to apply inputs to a logic circuit 2404 and may be configured to receive outputs of the logic circuit 2404 includes configuring the FPGA fabric to form a series of sequentially coupled latches to receive scan data. The sequentially configured latches 2408 are also coupled to the logic circuit 2404. The latches 2408 form a scan chain that is to receive test vectors that are scanned into it. Once the scan chain is fully loaded with test vectors (or one

scan sequence), the test vectors are clocked into circuitry 2404. Additionally, the output of circuitry 2404 is produced to the output scan chain where the output values are latched and may be produced for evaluation.

[0112] When the logic circuit 2404 produces the output of interest, this output is latched into the output scan chain and then scanned out as shown at 2416. This data may then be compared to the output that is expected to be produced by the logic circuit 2404 for the given input test vectors.

[0113] Alternatively, the FPGA fabric may be configured to at least partially evaluate the output test results to determine proper functional operation. The output scan chain may be different logic circuitry than the input scan chain logic circuitry or, alternatively, the same logic circuitry configured to receive output data. Moreover, these operations occur in conjunction with the use of fixed logic core processor scan chains to place the fixed logic core processor into a desired state and to observe the state of the fixed logic core processor after stimulation with particular inputs. While the foregoing example illustrates a possibility of using scan chains to establish and test fixed logic devices and circuits formed within the gasket logic, it should be understood that scan chains are not necessarily required for such tests.

[0114] Figure 25 is a functional and exemplary block diagram illustrating a variety of test modules configured within an FPGA fabric and how they are used to test gasket logic according to the present invention. The gasket logic 2504 under test includes data input lines, address input lines, control input lines, and at least one reset line. The gasket logic 2504 also includes data output lines and address output lines. As was described particularly with reference to Figures 21 and 22, test circuitry is employed to isolate the gasket logic so that the configured FPGA fabric may test the gasket logic.

[0115] As shown, a pair of 32-bit linear feedback shift

register (LFSR) counters 2512 stimulate the data input lines and address input lines of the gasket logic 2504. The LFSR counter 2512 generates a known pseudo-random sequence. Typical LFSR counters used herein during test are those that are able to generate the pseudo-random patterns for significant periods without repetition.

[0116] Additionally, a 14-bit Grey Code counter 2516 provides a Grey Code data pattern that stimulates the control inputs of the gasket logic 2504. A Grey Code data pattern is one in which only one bit of data is allowed to change from one generated number to a subsequently generated number. Grey Code counters are preferable for testing the control logic because they better simulate signal stability for input signals to a control module.

[0117] Finally, an 11-bit LFSR counter 2520 (containing decode logic as shown) stimulates the reset line(s) of the gasket logic 2504. The 11-bit LFSR counter (with the decode logic) 2520 produces a reset signal for the gasket logic 2504 after approximately 1,100 clock cycles in the described embodiment of the invention.

[0118] The combination of the inputs provided by the 32-bit LFSR counters 2512, the 14-bit Grey Code counter 2516, and the 11-bit LFSR counter 2520 fully stimulate the inputs of the gasket logic 2504. Based upon simulation results, the gasket logic 2504 will produce a particular output at each testing clock cycle. The outputs of the gasket logic 2504 are produced at the data lines and the address lines. These outputs are received by two 32-bit Cyclic Redundancy Check (CRC) modules 2508 configured in the FPGA fabric. Based upon expected outputs, approximately every 150,000 clock cycles in one described embodiment of the invention, the CRC modules 2508 will output a particular data pattern that may be used by the tester for examination for an expected data pattern. When the expected data pattern is not received as expected at a specified test point, e.g., the 150,000th clock cycle, the gasket logic 2504 has failed its testing. It is understood

that the above described embodiment is exemplary and may be modified as necessary or derived without departing from the inventive concepts disclosed herein.

[0119] Figure 26 is a functional and exemplary block diagram that illustrates a portion of the MISR circuitry used to receive the outputs from the gasket logic and to provide the output for comparison purposes. The structure of Figure 26 is configured in the FPGA fabric during gasket logic testing registers. The illustrated portion of the multiple input signature register (MISR) circuitry receives a plurality of inputs from the gasket logic 2604. These inputs are provided to a plurality of exclusive OR (XOR) gates 2608, each of which also receives as its second input the output of an adjacent latch (corresponding to an adjacent data line). These latches are shown generally at 2612. Note that for a 32-bit MISR circuit, 32 XOR gates and 32 latches, and a 4-Bit XNOR tapped off of bits 32, 22, 2 and 1 would be included in the described embodiment of the invention.

[0120] At each clock cycle, the XOR gates 2608 combine the value produced by the gasket logic 2604 with the data value of the adjacent latch. The resulting output produced by the XOR gates is then latched into the corresponding latch for use in the subsequent clock cycle/gasket logic data cycle. After a specified number of clock cycles, e.g., 150,000 in the described embodiment, the latches 2612 hold a unique data value. This unique data value may be converted to serial data via multiplexer 2616 and output to a tester conducting the test of the FPGA. The tester may then compare this unique data value to an expected/correct data value. Alternately, the expected/correct data value may be loaded into the MISR module as a data value from shift register 2620 and compared by logic circuitry with the value produced. More specifically, a comparator connected in place of multiplexer 2616 compares the values found in the shift register 2620 with the expected/correct value. The result of this comparison may then be output to the tester in place of

serial outputs from multiplexer 2616. It is understood the system of Figure 26 is functional in nature and explains the general concept of applying a signature function to sequentially generated outputs for functional evaluation. Any known design for applying a signature function (such as a CRC) may be used in place of that which is shown in Figure 26.

[0121] Figure 27 is a functional block diagram illustrating a configuration used for testing a fixed logic core processor embedded within an FPGA fabric according to the present invention. More specifically, an FPGA fabric 2700 includes an embedded fixed logic core 2704 that may be a microprocessor, a digital signal processor, an input/output device, or another fixed logic device. The particular fixed logic core described with reference to Figure 27 is an IBM PowerPC 405 that includes 412 input lines and 526 output lines.

[0122] During test, a scan chain (here, scan chain "0") is loaded into FPGA circuitry. Moreover, the core 2704 is placed into a known state using its ten scan chains and scan chain clocking inputs since core 2704 is a PowerPC 405. Then all, or a portion, of the 412 inputs are stimulated by the FPGA fabric and its scan chain (and test vectors) and the core 2704 is clocked. At a next clock cycle, a different set of inputs may be applied and the core 2704 may be clocked again. This procedure is repeated for a plurality of clock cycles. After the particular testing sequence is completed, the state of the core 2704 is retrieved using its ten scan chain outputs, and some or all of the plurality of outputs are retrieved and compared to expected/correct values derived from simulation. This test sequence will typically be repeated a number of times to obtain substantial fault coverage of the core 2704.

[0123] Thus, according to the present invention, test circuitry and communication paths are configured in the FPGA fabric 2700 that work in conjunction with a tester to test

the core 2704. After configuration of the FPGA fabric 2700 for testing purposes, the test configuration includes an input configuration 2702, an output configuration 2708, and a plurality of off-chip connections. These off-chip connections allow the tester to provide control signals such as A, B, and C clocks to the input configuration 2702, the core 2704, and the output configuration 2708. These off-chip connections also allow the tester to apply SCAN IN [1:10] data and to receive SCAN OUT [1:10] data. In addition to this FPGA fabric 2700 configuration, the gasket logic surrounding the core 2704 is also placed into a state that provides direct access by the FPGA fabric 2700 to the core 2704 (see description of Figure 21 or 22).

[0124] The input configuration 2702 includes a plurality of logic blocks, each of which forms a portion of a scan chain that provides one or a plurality of the 412 inputs to the core 2704. One particular structure of such logic blocks is illustrated with reference to Figure 28A. The input scan chain of the input configuration 2702 is connected to receive SCAN IN [0] data that is provided by the tester.

[0125] The FPGA fabric 2700 is also configured to provide access to the ten input scan chains of the core 2704, SCAN IN [1:10] and to the ten output scan chains of the core 2704, SCAN OUT [1:10]. With this configuration of the FPGA fabric 2700, a tester having access to the input/output of the FPGA fabric 2700 has direct access to the scan chains of the core 2704.

[0126] The output configuration 2708 receives some or all of the 526 outputs of the core 2704, latches the outputs upon direction of the tester coupled to the FPGA fabric 2700, and scans out the latched data as SCAN OUT [0] that is received by the tester. The structure of a plurality of logic blocks making up the output configuration 2708 may be similar to the structure described with reference to Figure 28A.

[0127] Additionally, the A clock, B clock and C clock signals described in relation to Figure 28 are also provided

to the fixed logic core processor by way of the FPGA and the gasket logic formed around the fixed logic core processor. The latches used for scan chains may be the same set of latches or different sets of latches, one for each function.

[0128] Figure 28A illustrates a latch shift register 2804 that is used in one embodiment of the logic blocks of the input configuration 2702 of Figure 27. The latch shift register 2804 includes a pair of modules (latches) 2808 and 2812. The first latch 2808 includes a multiplexer 2808C, an OR gate 2808A, and a latch 2808B that performs a desired latching function.

[0129] The first latch 2808 is coupled to receive a scan (test) signal (S), a data signal (D), an A clock (A), and a C clock (C). In the described embodiment, the clock signals are broadcast signals. The Multiplexer 2808C is used to select between two data sources (S and D) and is clocked by an input that is coupled to the output of OR gate 2808A. Thus, latch 2808B is clocked upon the clock pulse of either one of the A clock or the C clock to receive either the test signal (S) or data signal (D).

[0130] The second slave latch 2812 is a slave latch that receives the output of latch 2808B. Slave latch 2812 is driven by the B clock, however, and therefore its latching can be configured to occur independently of the events and clocks that drive latch 2808B. As is shown in Figure 28B, a "B" clock pulse is set to occur after either an "A" clock or a "C" clock pulse to enable the slave latch to capture data regardless of the source (regardless of whether the data is real or is test data).

[0131] More specifically, the output of latch 2808B is coupled to the data input of slave latch 2812. The clock input of slave latch 2812 is also coupled to receive the B clock signal. Accordingly, upon the receipt of the B clock pulse from the third clock source, the output value of latch 2808B is input into latch 2812 and latched to produce the corresponding bit (or multiple bits) on its output line.

This data is then applied to the PowerPC core.

[0132] During a series of scanning operations, the output of the first latch 2808B is received by the input of an adjacent logic block 2804. This structure allows a scan vector to be input into a plurality of these circuits that form a scan chain for input to the fixed logic PowerPC core. Once all of the latches are loaded with input test data, the test data (test vectors) may be produced to the DUT for test by way of output 2816. Furthermore, this structure allows the logic state of PowerPC core to be captured and scan out to be evaluated, also by way of output 2816.

[0133] Figure 29 is a functional block diagram illustrating the use of a scan chain in an FPGA fabric to test an embedded device according to one aspect of the present invention. As mentioned before, a plurality of test configurations may be defined, each of which is for performing a specified test. Thus, one particular scan chain is for establishing logic states and circuit configuration within the FPGA fabric to support testing of the embedded device, as shown in the functional block diagram of Figure 29.

[0134] As may be seen, a plurality of shift registers within the FPGA fabric are configured as a scan chain 2904 to facilitate the transmission of test vectors into the embedded device and to facilitate receiving data with test results through the scan chains from the embedded device, among other functions. In the described embodiment of the invention, different scan chains are loaded into the FPGA to accomplish these different results.

[0135] The gasket logic portion comprises, in the described embodiment of the invention, approximately 600 pins. The Power PC chip, on the other hand, comprises nearly 1,000 pins. Accordingly, the use of the scan chains to establish circuit conditions is important to significantly reduce the number of communication paths that must be established to the device under test through the FPGA fabric

and the gasket logic and to reduce the processing steps to set up a desired set of logic states to support test conditions.

[0136] Continuing to refer to Figure 29, scan chain 2904 therefore facilitates the creation of communication channels (through scan in chain 0), shown generally at 2912, for transmitting test data (represented by an "x" in the circles that represent the scan latches) directly into fixed logic device 2908 and then out of fixed logic device 2908 at the communication channels shown generally at 2916 for receiving test data outputs from the fixed logic device 2908 (through scan out chain 0). Additionally, configured path 2906 establishes the communication paths through the FPGA fabric 2900 to defined communication channels of the fixed interfacing logic circuitry for transmitting scan chains 1-10 for the embedded core device, and for receiving output scan chains 1-10 with internal processing results from it.

[0137] As is understood, the scan chain 2904 is particular to the type of embedded fixed logic device 2908. Stated differently, each type of embedded device requires different couplings for stimulating the device, as well as for transmitting and receiving communication signals through established communication channels. For example, the fixed logic device 2908 (an IBM PowerPC) requires channels for receiving test data and for outputting test data, as well as communication channels for receiving and outputting scan chains 1-10.

[0138] Figure 30 is a functional schematic diagram of an FPGA constructed according to one embodiment of the invention. More particularly, an FPGA fabric includes gasket logic that further includes a plurality of embedded core processor blocks. In the described embodiment, an FPGA fabric 3000 that includes four core processor blocks 3004, each of which is an embedded device of the same type in the described embodiment.

[0139] The FPGA 3000 is formed to receive scan chains 3008

to facilitate testing of the embedded devices. Each of the core processor blocks further is coupled to receive test data through a plurality of communication paths, shown generally at 3012, that are created by logic states of the logic circuitry of the FPGA as well as by the gasket logic. Additionally, each of the core processors 3004 is coupled to receive scan chains for internal tests through a plurality of communication paths 3014. While the example of Figure 30 illustrates test data being received from an external source, an alternate approach includes having the FPGA 3000 generate the test data.

[0140] While not specifically shown, it is understood that the circuitry further defines communication paths for the transmission of a plurality of clock signals to each of the scan chains. For example, the A clock, B clock and C clock for the shift registers that receive the scan chains, as described herein in relation to Figure 27, are transmitted through communication paths not shown herein.

[0141] Given the synchronous operation that necessarily occurs responsive the receipt of the clock pulses by each of the embedded fixed logic core processor blocks 3004 , as well as the parallel test data streams, each of the core processor blocks 3004 produces similar outputs at or about the same instant, assuming proper circuit operation.

[0142] The outputs of each core processor block 3004 are produced to a comparator 3016 where their values may be compared to each other and to an expected output value received from a memory 3020 that is used for storing the expected value. If the expected value is the same as the output value of all four core processors, the comparator 3016 produces a "pass" indication on output logic path 3024. Comparators such as comparator 3016 are known.

[0143] In an alternate embodiment of the invention, the outputs of the core processor blocks 3004 are merely compared to each other and are not compared to a stored value from a memory, which stored value reflects an expected output value.

For this alternate embodiment, the assumption is made that the probability all four processors 3004 having a failure that produces the same result is so low that the risk of such a failure going undetected is acceptable. In the described embodiment, however, comparing the actual outputs to an expected value precludes such a possibility, however slight.

[0144] The system of Figure 30 illustrates an embodiment of the invention wherein the core processor blocks 3004 all contain the same type of core processor as an embedded device. If, for example, the embedded devices are of a different type, as will be the case for some embodiments, one of several approaches may be pursued. First, if a tester has the capacity to generate test signals for multiple devices at the same time (meaning it has adequate input and output capacity for supporting the multiple devices), then scan chains particular to the device being tested are loaded around the core processor blocks and input data unique to the embedded device is produced thereto. Alternatively, scan chains are loaded only for the device under test and corresponding data is produced thereto. Feasibly, four tests would be performed for a system similar to that shown in Figure 30 if each embedded device is different.

[0145] Figure 31 is a flow chart illustrating a method for testing an embedded core device according to one embodiment of the present invention. As has been described before, the FPGA disclosed herein includes an embedded cored device that is surrounded by interfacing logic that is for interfacing the embedded cored device to the FPGA fabric, among other purposes. A first step in testing such an embedded core device, therefore, is to configure the FPGA into a mode for performing the testing of the embedded core device (step 3104). As a part of configuring the FPGA for the embedded core test, communication paths in the interfacing logic must be created or made electrically present (step 3108). These communication paths are useful for producing scan chains, if necessary, to the embedded core device, as well as for

producing test data. Thus, the next step includes producing test vectors within the scan chains to the embedded core device (step 3112). Additionally, the inventive process includes producing test data and conducting the test data through the interfacing logic to the inputs of the core device (step 3116). Once the test has been set up through the scan chains and through the data being produced to the device under test, a clock pulse is generated to prompt the embedded device to process the data and produce resulting outputs. Accordingly, the invention includes, after generating the clock pulse, receiving output scan chains, or at least one output scan chain, from the embedded core device (step 3120). The output scan chains are then conducted to an external tester (step 3124) where it may determine whether electrical integrity or functional integrity exists (step 3128).

[0146] Figure 32 is a flow chart illustrating a method for testing at least one embedded circuit within an FPGA according to one embodiment of the present invention. Referring now to Figure 32, the FPGA must be configured to isolate at least one embedded circuit (step 3204). As was described herein, the test circuitry is not only created by the creation of select configurations of the FPGA fabric, but also by causing designated circuit elements to become electrically present while the FPGA is in a test mode of operation. For example, multiplexers are formed within the interfacing logic of the FPGA, which multiplexers are for isolating circuit devices and for facilitating the delivery and receipt of test signals there to and there from. Thus, once a device has been isolated, a scan chain is scanned in for the device if the device is of a type that utilizes scan chains for test purposes (step 3208). For example, if the device under test is a PowerPC, then its set of scan chains would be connected to for test purposes. On the other hand, if the device under test were merely a circuit element that is a part of the fixed interfacing logic, then no scan chain

would be loaded into the device. Once the FPGA and test circuitry are configured to perform tests, then test signals are conducted to the at least one embedded circuit (step 3212). As was described previously, a plurality of devices may be embedded into the FPGA and surrounded by fixed interfacing logic. Accordingly, in such an embodiment, the step of conducting test signals to at least one embedded circuit would include conducting test signals to all of the embedded test circuits.

[0147] Once the test signals have been conducted to the at least one embedded circuit, a clock pulse is produced to each of the at least one embedded circuits to prompt them to process the test signals (step 3216). Thereafter, the outputs are received from the at least one embedded circuit (step 3220) and the system determines whether the device passed or failed (step 3224).

[0148] Figure 33 is a flow chart illustrating a method for testing a specified circuit element or module formed within the fixed interfacing logic of an FPGA according to one embodiment of the present invention. Referring now to Figure 33, the first step is to configure the FPGA into a specified test mode (step 3304). For example, the specified test mode can be one that is for testing an embedded core device or, as here, for testing a specified circuit element or module of the fixed interfacing logic. Thereafter, the specified circuit element or module that is to be tested is isolated for test purposes (step 3308). By isolated, what is meant is that test multiplexers are made electrically present to define new and temporary communication paths through the fixed interfacing logic that are used for test purposes. Thus, in addition to isolating the specified module, the communication paths for performing the tests are created through the fixed interfacing logic, as well as through the FPGA fabric because of its test configuration (step 3312).

[0149] The preceding steps relate to preparing the FPGA to perform at least one specified test. Thus, once the FPGA

is ready to perform the test, test signals are generated to the fixed interfacing logic through the FPGA fabric, and then in the fixed interfacing logic, through a test mode and communication path (step 3316). The communication path referred to in step 3316 is one that was created for test purposes or made electrically present for test purposes within the fixed interfacing logic. Once a clock pulse has been generated to prompt the device to process the test signals, the method includes receiving and evaluating the response from the device in the fixed interfacing logic to determine whether it has passed or failed (step 3320). Thus, the final step includes making the determination of whether the circuit element of the fixed interfacing logic passed or failed the test (step 3324).

[0150] The inventive method and apparatus disclosed herein are particularly advantageous in that they provide a capability for testing a device that is embedded within an FPGA fabric. While the invention is susceptible to various modifications and alternative forms, specific embodiments thereof have been shown by way of example in the drawings and detailed description. It should be understood, however, that the drawings and detailed description thereto are not intended to limit the invention to the particular form disclosed, but, on the contrary, the invention is to cover all modifications, equivalents and alternatives falling within the spirit and scope of the present invention as defined by the claims. As may be seen, the described embodiments may be modified in many different ways without departing from the scope or teachings of the invention.